Robust Optimization for Topological Surface Reconstruction

ROEE LAZAR, Weizmann Institute of Science
NADAV DYM, Weizmann Institute of Science
YAM KUSHINSKY, Weizmann Institute of Science
ZHIYANG HUANG, Washington University in St. Louis
TAO JU, Washington University in St. Louis
YARON LIPMAN, Weizmann Institute of Science

Surface reconstruction is one of the central problems in computer graphics. Existing research on this problem has primarily focused on improving the geometric aspects of the reconstruction (e.g., smoothness, features, element quality, etc.), and little attention has been paid to ensure it also has desired topological properties (e.g., connectedness and genus). In this paper, we propose a novel and general optimization method for surface reconstruction under topological constraints. The input to our method is a prescribed genus for the reconstructed surface, a partition of the ambient volume into cells, and a set of possible surface candidates and their associated energy within each cell. Our method computes one candidate per cell so that their union is a connected surface with the prescribed genus that minimizes the total energy. We formulate the task as an integer program, and propose a novel solution that combines convex relaxations within a branch and bound framework. As our method is oblivious of the type of input cells, surface candidates, and energy, it can be applied to a variety of reconstruction scenarios, and we explore two of them in the paper: reconstruction from cross-section slices and iso-surfacing an intensity volume. In the first scenario, our method outperforms an existing topology-aware method particularly for complex inputs and higher genus constraints. In the second scenario, we demonstrate the benefit of topology control over classical topology-oblivious methods such as Marching Cubes.

CCS Concepts: • **Computing methodologies** → *Mesh models*;

Additional Key Words and Phrases: surface reconstruction, convex optimization, topological constraints, connectivity constraints

ACM Reference Format:
Roee Lazar, Nadav Dym, Yam Kushinsky, Zhiyang Huang, Tao Ju, and Yaron Lipman . 2018. Robust Optimization for Topological Surface Reconstruction. *ACM Trans. Graph.* 11, 1, Article 46 (January 2018), 10 pages. https://doi.org/http://dx.doi.org/10.1145/8888888.7777777

1 INTRODUCTION

Surface reconstruction from incomplete data (e.g., images, volumes, points, curves, etc.) is one of the central topics in computer graphics. To be useful for downstream tasks, the surfaces need to satisfy application-dependent requirements. One class of such requirements concerns the surfaces' topology, including its connectedness and genus. Topological requirement often arises in biology and medicine, as many natural objects (e.g., anatomical structures) have a known topology; for example the corn roots whose reconstruction is shown in Figure 1 are known to be connected, genus zero surfaces. Geometric processing tasks, such as parameterization and shape matching, also prefer shapes with simple and consistent topology.

Fig. 1. Topologically correct, connected, genus zero reconstruction of a corn root (right) from cross sections (left).

Despite the extensive research on surface reconstruction, few works have addressed the topological correctness of their output. The scarcity is in part due to the fact that topology is an inherent property that is invariant of geometric deformations. Hence it is difficult to change topology (without creating adverse impacts on geometry), and even more difficult to enforce a prescribed topology. To satisfy topological requirements, existing reconstruction methods either rely on user interaction, which can be tedious and not suited for batch processing, or resort to post-processing topology repair, which generally has no knowledge of the input data from which the surface is created and hence can make incorrect decisions.

In this work, we propose a solution for automatic topology-aware reconstruction that outputs topologically correct surfaces without the need for post-processing repair. Our method is inspired by the recent works [Huang et al. 2017; Zou et al. 2015] which are among the first topology-aware reconstruction methods. The methods are designed for the problem of surface reconstruction from 2D cross-section curves, and take a divide-and-conquer approach. Given a partitioning of space into "cells" by the cross-section planes, they first enumerate multiple candidate surfaces within each cell that connect the curves on the cell boundary. They then solve a challenging combinatorial optimization problem to select one candidate per cell so that their union surface has the prescribed topology while minimizing a geometric energy. These methods are effective in meeting the topological requirements and are rather efficient for simple data sets consisting of a few planes. However, a simple optimization method (dynamic programming) is used in these methods. Although

Authors' addresses: Roee Lazar, Weizmann Institute of Science; Nadav Dym, Weizmann Institute of Science; Yam Kushinsky, Weizmann Institute of Science; Zhiyang Huang, Washington University in St. Louis; Tao Ju, Washington University in St. Louis; Yaron Lipman, Weizmann Institute of Science.

0730-0301/2018/1-ART46 $15.00
https://doi.org/http://dx.doi.org/10.1145/8888888.7777777

optimal, dynamic programming scales poorly with the complexity of the inputs, such as the number of planes, complexity of curves on each plane, and the genus constraint.

Our work aims at developing an efficient, effective and general-purpose combinatorial optimization method, which can enable topology aware reconstruction not only from cross-sections but also from any other data where the divide-and-conquer approach is applicable. We formulate the candidate-selection task as an integer programming problem, and show that the topological constraints on the consistency, genus, and connectivity of the surface can be translated into convex constraints in an integer valued vector x. By convexifying the integer constraints we obtain convex relaxations for the topology-aware reconstruction problem which leads to tight lower bounds for the optimal value of the problem. These tight lower bounds are exploited to solve the combinatorial optimization problem efficiently in a small number of branch and bound iterations.

We explore two reconstruction scenario that utilize our new optimization method. For the problem of reconstructing from cross-sectional curves, our results show that our method is significantly more scalable than dynamic programming, often finding optimal results several orders of magnitude faster. The scalability allows our algorithm to process much more complex inputs and higher genus constraints that exceed the capability of methods like [Huang et al. 2017; Zou et al. 2015]. In a second problem, we develop a topology-aware iso-surfacing method for grid data. Unlike conventional methods (e.g., Marching Cubes) that determines the patch structure within each grid cell solely based on local information (e.g., signs at the grid points), our method is guided by global topological constraints and hence is more likely to produce a topologically correct iso-surface.

Of independent interest is the formulation of the connectivity constraint on the surface as a convex constraint , which we reduce to the problem of constraining an appropriate graph to be connected. We explore two convex formulations of graph connectivity: edge-connectivity and algebraic connectivity. We show edge-connectivity is strictly tighter (*i.e.*, produce better lower bounds) than algebraic connectivity and propose a cutting-plane-like algorithm to efficiently incorporate edge-connectivity constraints by iteratively solving a chain of linear programs. We use algebraic connectivity as a safe-guard for rare cases where convergence of the cutting plane algorithm is slow.

2 PREVIOUS WORK

Topology control in surface reconstruction. Topological control can be achieved by either user interaction [Sharf et al. 2007; Yin et al. 2014] or template fitting [Bazin and Pham 2007; Zeng et al. 2008]. However, these methods are limited to the availability of human resource or template structures. An alternative strategy is removing topological errors from on an existing surface [Ju et al. 2007; Wood et al. 2004] (see more in-depth discussion in the survey [Attene et al. 2013]). However, as these post-processing methods generally have no knowledge of the data from which the surface is created, they could make repair decisions that result in undesirable geometry that deviate from the inputs (*e.g.*, surface no longer interpolating the input point clouds or cross-section curves).

Only a few works have attempted to incorporate topology constraints within an automatic reconstruction algorithm. Sharf et al. [Sharf et al. 2006] proposes an advancing-front method, designed for point cloud inputs, which allows control over the genus of the result. A divide-and-conquer approach is used in several works for reconstructing 2D curves [Zhou et al. 2014], and, more recently, 3D surfaces [Huang et al. 2017; Zou et al. 2015]. These algorithms first enumerate possible candidates within each cell of a spatial subdivision and measure the suitability of each candidate as an energy. Given a target topological goal (*e.g.*, number of connected components and/or genus), they solve a combinatorial optimization problem that selects one candidate per cell to achieve the goal while minimizing the total energy. While these methods differ in the dimensionality of the problem, the enumeration of candidates, and the definition of energy, they all use dynamic programming (DP) in the optimization step. Starting from an initial cell, DP sequentially adds adjacent cells, keeping track of all topological possibilities as the algorithm progresses. In each stage topological possibilities are discarded if they have a larger number of handles than the prescribed genus. The DP algorithm will eventually find the globally optimal solution, but has worse case exponential time and space complexity.

Reconstruction from cross-sections. Surface reconstruction from cross-section slices has been extensively studied for the past few decades (see more in-depth reviews in [Bermano et al. 2011; Zou et al. 2015]). While earlier methods are specialized for closed curves on parallel cross-sections, more recent methods can handle non-parallel, intersecting planes [Boissonnat and Memari 2007] and even multi-labelled domains on each plane [Barequet and Vaxman 2009; Bermano et al. 2011; Liu et al. 2008]. The majority of these methods rely on the spatial subdivision by the cross-section planes, and focuses on surfacing within each cell to interpolate the curves on the cell boundary. However, unlike [Huang et al. 2017; Zou et al. 2015], most existing methods create a single surface within each cell that is determined locally, with no guarantee on the topology of their union. An exception to this rule is the algorithm presented in [Amini et al. 2013] which is guaranteed to achieve a correct topological reconstruction for a sufficiently dense choice of intersection planes. In contrast our algorithm is applicable for scenarios in which only a sparse input is given.

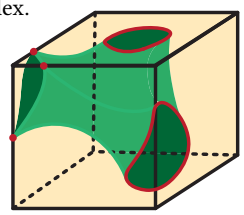
Iso-surfacing. Polygonalizing iso-surfaces is another well-studied problem in computer graphics and visualization. We refer readers to a recent survey [De Araújo et al. 2015] on this topic. Arguably, the most successful iso-surfacing method is the Marching Cubes algorithm [Lorensen and Cline 1987], which creates triangles within each cubic cell based on the signs at the cell corners. Significant effort has been made to improve Marching Cubes, particularly for capturing the topology of the analytical iso-surface defined by some interpolant (e.g., trilinear) of the values at the cell corners [Chernyaev 1995; Cignoni et al. 2000; Custodio et al. 2013; Velasco et al. 2008]. However, all existing grid-based iso-surfacing algorithms determine a unique choice of surface within each grid cell using only local information (e.g., signs and values at cell corners). On the other hand, our algorithm is guided by a global topological constraint, and explores multiple candidates per cell to seek a surface that meets the constraint.

Graph connectivity. Our algorithm reduces the surface connectivity problem to a graph connectivity problem, and then enforces graph connectivity using convex constraints. Several papers in different fields have used semi-definite programming (SDP) to enforce positive algebraic connectivity on graphs. [Qian et al. 2014] use SDPs to optimize a given energy over all connected subgraphs of a given graph. [Das and Mesbahi 2005] use SDP relaxations to optimize over the set of k-connected graphs. In [Ghosh and Boyd 2006] SDP relaxations are used to find k edges to add to the graph such that the algebraic connectivity of the graph is maximized. The latter paper seems to be the only one to propose a lower bound for the algebraic connectivity of a graph, a lower bound which we use as well. This bound is based on the seminal paper by Fiedler on bounding the algebraic connectivity [Fiedler 1973], and establishing connections between algebraic connectivity, vertex connectivity and edge connectivity. In our case we show theoretical and empirical evidence to the fact that enforcing edge connectivity can significantly enhance the accuracy of graph connectivity relaxations.

3 METHOD

3.1 Problem definition.

We consider a cell complex $C = \{C_i\}_{i=1}^n$ decomposition of a given domain $\Omega \subset \mathbb{R}^3$. That is, a collection of (not necessarily bounded) convex polyhedral cells C_i, such that $\Omega = \cup_i C_i$, and any two cells either do not intersect, or their intersection is a sub-polyhedron (vertex, edge, or a polyhedral face). We assume that we are given a sampling of the surface on the cell boundaries, represented by segments $\Gamma = \{\gamma_k\}_{k=1}^m$, *e.g.*, the red curves and points in the inset. Note these are not necessarily entire intersections of surfaces with cells' boundaries. Each segment is connected and resides in one or more polyhedral faces of the cell complex.



In each cell C_i we are looking for a surface patch S_i out of a collection of c_i (topologically different) possible valid triangulated surface patches $\{S_{i1}, S_{i2}, \ldots, S_{ic_i}\}$ interpolating the segments γ_k which reside in the boundary of this cell. By valid surface patches we mean that each S_{ij} is a surface, whose intersection with the boundary of C_i is a finite collection of closed curves. By interpolating the segments we mean that every segment γ_k of a cell is contained in every candidate surface patch. Furthermore, we require that the intersection of every surface patch S_{ip} with the sub-cells of its cell (*i.e.*, the cell itself, or faces, edges or points in the cell) contains at-least a part of a segment γ_k in every connected component of the intersection. For example, the inset shows in green a potential surface patch candidate S_{ip} that interpolates the red segments in a cell.

The task is to choose a single surface patch per cell such that a fair connected manifold surface S of some prescribed topology is achieved.

Problem 1. *Find a manifold compact surface triangulation $S = (V^S, E^S, F^S)$, where $\Gamma \subset S$, by choosing a surface patch per-cell so that S is connected, matches a prescribed topology g_0 and minimizes an appropriate convex energy.*

3.2 Approach

The first step of our approach for solving Problem 1 is formalizing it as an optimization problem.

We begin by defining the unknowns. Per cell C_i, $i = 1, \ldots, n$ we define an indicator unknown vector $x_i = [x_{i1}, x_{i2}, \ldots, x_{ic_i}]$, responsible for choosing the surface patch $S_i \in \{S_{i1}, \ldots S_{ic_i}\}$ at cell C_i out of c_i possible predefined (topologically different) options,

$$\sum_{j=1}^{c_i} x_{ij} = 1 \tag{1a}$$

$$x_{ij} \in \{0, 1\} \tag{1b}$$

The reconstructed surface is defined to be

$$S = S(x) = \bigcup_{i=1}^{n} \sum_{j=1}^{c_i} x_{ij} S_{ij}.$$

Problem 1 can be formulated as the optimization problem:

$$\min_x \quad E(x) \tag{2a}$$

$$\text{s.t.} \quad x \text{ satisfies (1)} \tag{2b}$$

$$S(x) \text{ is well defined across cells} \tag{2c}$$

$$S(x) \text{ is of genus } g_0 \tag{2d}$$

$$S(x) \text{ is connected} \tag{2e}$$

We show that the topological constraints (2c)-(2e) can be formulated as convex constraints in the variable x. Once this is achieved, Problem 1 is an integer program that can be convexified by simply relaxing the $0/1$ constraint to the constraint

$$0 \le x_{ij} \le 1. \tag{3}$$

Our experiments show that this convex relaxation is very tight, and in most cases can be used to globally optimize Problem 1 in a small number of branch and bound iterations.

3.3 Convex formulation of topological constraints

In this section we consider Problem (2) (with integer variables x) and formulate (2c), (2d), (2e) as convex constraints in x.

Consistency. The constraint (2c) amounts to the requirement that if C_i and C_k are two adjacent cells such that $B = B(i,k) = C_i \cap C_k$ is non-empty, then the surface patches selected for the two cells will agree on their common boundary. The surface patches define a finite number of possibilities for the intersection of the reconstructed surface S with B, which we denote by b_ℓ, $\ell = 1, \ldots, L$. For x satisfying (1), the reconstructed surface $S(x)$ will be consistent if and only if x satisfies the linear constraints

$$\sum_{j:\, S_{ij} \cap B = b_\ell} x_{ij} = \sum_{j:\, S_{kj} \cap B = b_\ell} x_{kj}, \quad \text{for all } i, k, \ell. \tag{4}$$

Genus constraint. The constraint (2d) is equivalent to requiring the Euler characteristic of $S(x)$ to be $\chi_0 = 2 - 2g_0$. If a surface patch S_{ij} is selected, and v is a vertex in this surface patch which is on the boundary of the cell C_i, then by the consistency constraints v will also be a member of all surface patches selected for the other

cells which contain v. Denoting by $n_v(v)$ the number of cells which contain v, we have that the number of vertices in $S(x)$ is given by

$$|V| = \sum_{ij} x_{ij} \left(\sum_{v \in S_{ij}} n_v^{-1}(v) \right).$$

Using a similar argument to count the edges and faces of $S(x)$, we obtain that (2d) can be phrased using the linear constraint

$$\chi_0 = \sum_{ij} x_{ij} \left(\sum_{v \in S_{ij}} n_v^{-1}(v) - \sum_{e \in S_{ij}} n_e^{-1}(e) + \sum_{f \in S_{ij}} n_f^{-1}(f) \right), \quad (5)$$

where the r.h.s. is the Euler formula for the reconstructed surface $S(x)$. Note that in practice since we require surface patches to be valid, we always have $n_f(f) = 1$.

Connectivity. For a fixed integer valued vector x, we define a graph $G(x) = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E} = \mathcal{E}(x)$ so that $S(x)$ is a connected surface if and only if $G(x)$ is a connected graph. Each vertex $u_k \in \mathcal{V}$ of the graph corresponds to an input segment from $\Gamma = \{\gamma_k\}_{k=1}^m$ (u_k can be imagined as a point in γ_k), therefore $|\mathcal{V}| = m$. The edges $\mathcal{E}$ of the graph are defined by considering each surface patch S_{ij} for which $x_{ij} = 1$, and connecting pairs of vertices $u, u_0 \in \mathcal{V}$ if they belong to the same connected component of S_{ij}. We prove the following theorem in the appendix:

Theorem 3.1. *$S(x)$ is connected if and only if $G(x)$ is connected.*

The connectivity of $G(x)$ can be enforced algebraically in two related, but different ways: using the *edge connectivity* and the *algebraic connectivity*. Both are functions of the Laplacian $L(x)$ of $G(x)$.

To define the Laplacian of $G(x)$ as a function of x, consider for each S_{ij} the sub-graph $G_{ij} = (\mathcal{V}, \mathcal{E}_{ij})$ obtained by selecting only the edges induced by this surface patch. The Laplacian $L^{ij} \in \mathbb{R}^{m \times m}$ of G_{ij} is a constant (*i.e.*, independent of x) matrix defined by

$$L_{kl}^{ij} = \begin{cases} \deg(u_k) & u_k = u_l \in S_{ij} \\ -1 & u_k \neq u_l \text{ are connected by } S_{ij} \\ 0 & \text{otherwise} \end{cases}.$$

where $\deg(u_k)$ is the number of vertices attached to u_k in G_{ij}. Note that by construction $L^{ij}\mathbf{1} = 0$. The Laplacian matrix of $G(x)$ is then

$$L(x) = \sum_{ij} x_{ij} L^{ij}.$$

Edge connectivity. Note that $G(x)$ is disconnected if and only if there is some $J \subset \mathcal{V}$ such that there are no edges connecting J and $V \setminus J$. Accordingly $G(x)$ is connected if and only if for all $J \subset \mathcal{V}$

$$e(x, J) \equiv \sum_{q \in J, r \notin J} -L_{qr}(x) \geq 1, \quad (6)$$

Or equivalently if

$$e(x) \geq 1, \quad (7)$$

where $e(x)$ is the *edge connectivity* of $G(x)$ which is defined as

$$e(x) = \min_{J \subset \mathcal{V}} e(x, J). \quad (8)$$

We note (6) is convex (in the variables x_{ij}) since it is defined by $2^{m-1} - 1$ linear inequality constraints.

Algebraic connectivity. A tractable convex connectivity constraint can be achieved by using a classical result in spectral graph theory [Chung 1997] which states that the graph $G(x)$ is connected iff $\lambda_2(x) > 0$, where $\lambda_2(x)$ is the second smallest eigenvalue of the Laplacian $L(x)$. This motivates the definition of $\lambda_2(x)$ as the *algebraic connectivity* of the graph $G(x)$. The relation of the algebraic and edge connectivity are given by the next inequality [Fiedler 1973]

$$\lambda_2(x) \geq e(x)\bar{\lambda}, \text{ where } \bar{\lambda} = 2\left(1 - \cos\frac{\pi}{m}\right) \quad (9)$$

and since for connected graphs $G(x)$ we have that $e(x) \geq 1$ we see that $G(x)$ is connected if and only if $\lambda_2 \geq \bar{\lambda}$. As shown in [Ghosh and Boyd 2006], this constraint is convex and can be enforced by a positive-semidefinite constraint: Since the minimal eigenvector of the Laplacian is always a constant vector, $\lambda_2(x) \geq \bar{\lambda}$, iff

$$L(x) \geq \bar{L} = \bar{\lambda}\left(I - \frac{1}{m}\mathbf{1}\mathbf{1}^T\right). \quad (10)$$

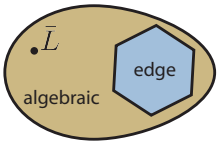
3.4 Convex relaxations

We saw that Problem 1 can be formulated as the integer programming problem of optimizing the energy E over all 0/1 solutions satisfying a collection of convex constraints. We will relax this problem by replacing the 0/1 constraint with its convex hull (3). For 0/1 variables x both connectivity constraints, *i.e.*, edge (7) and algebraic (10) are equivalent. However when relaxing the 0/1 constraints to (3), the edge-connectivity is in-fact tighter as can be understood from (9) which holds also for weighted graphs. Our convex relaxation is therefore

$$\min_x \quad E(x) \quad (11a)$$

$$\text{s.t.} \quad x \text{ satisfies (1a), (3), (4), (5)} \quad (11b)$$

$$G(x) \text{ is edge connected (7)} \quad (11c)$$



We denote this convex optimization problem P_{top}. To further quantify the gap between edge connectivity and algebraic connectivity, consider the matrix $\bar{L}$ from (10): a direct computation shows that its edge connectivity is $e(\bar{L}) = O(m^{-2})$ while its algebraic connectivity is $\lambda_2(\bar{L}) = \bar{\lambda}$; *i.e.*, the corresponding weighted graph is algebraic connected but pretty far from being edge connected. Another consequence is that algebraically disconnected graphs can be made algebraically connected by changing the edge weights by $O(m^{-2})$ (the magnitude of off-diagonal elements in $\bar{L}$). Our comparison of algebraic and edge connectivity is illustrated in the inset, and is summarized in the following Theorem (proved in the appendix): Let $\mathcal{L} = \mathcal{L}(m)$ be the set of laplacians of weighted graphs, that is,

$$\mathcal{L} = \{L \mid L = L^T, L\mathbf{1} = 0, \text{ and } L_{qr} \leq 0 \text{ for all } q \neq r\}.$$

Note that $\mathcal{L}$ is a convex cone, and in all of the relaxations we consider $L(x) \in \mathcal{L}$. In the following we set

$$\|L\|_1 = \sum_{ij} |L_{ij}|.$$

Theorem 3.2. *There is a constant $c > 0$ such that*

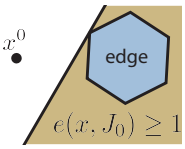
(1) *For all $L \in \mathcal{L}$, if $e(L) \geq 1$ then $\lambda_2(L) \geq \bar{\lambda}$.*
(2) *There exists an $L \in \mathcal{L}$ such that $\lambda_2(L) \geq \bar{\lambda}$ but $e(L) \leq cm^{-2}$.*
(3) *For all $L_0 \in \mathcal{L}$, there exists an $L \in \mathcal{L}$ such that $\lambda_2(L) \geq \bar{\lambda}$ and $\|L - L_0\|_1 \leq 2cm^{-1}$.*

Relaxation chain. Our goal is to solve P_{top} to achieve a good lower bound (and hopefully an exact solution) for (2). Since the edge connectivity constraint (7) cannot be formulated efficiently we use a variant of the *cutting plane* method to solve P_{top}. The idea is to completely remove the edge connectivity constraint (7) from P_{top}, solve the resulting linear program and iteratively improve the solution, as follows. Given a current solution x^0 that does not satisfy the edge connectivity constraint, namely

$$e(x^0) < 1$$

a *cutting plane* is found which separates x^0 and all edge connected x satisfying $e(x) \geq 1$. Then a linear inequality forcing x to lay on the side containing the edge-connected x is added to the relaxation and it is solved again to produce x^1. If $e(x^1) \geq 1$ the algorithms terminates (it found the optimal solution to P_{top}), otherwise it continues to add cutting plane inequalities.

Finding a cutting plane for x^0 amounts to computing the global min-cut J_0 of the weighted graph $G(x^0)$, *i.e.*, $e(x^0, J_0) = e(x^0)$. This can be done using the Stoer-Wagner algorithm [Stoer and Wagner 1997] which computes the global min cut in $O(|\mathcal{V}||\mathcal{E}| + |\mathcal{V}|^2 \log |\mathcal{V}|)$ time . The linear inequality $e(x, J_0) \geq 1$ would then exclude x^0 from the feasible edge-connected set as illustrated in the inset. The effectiveness of even a single cutting plane constraints in comparison with algebraic connectivity constraints is illustrated by comparing the third claim in Theorem 3.2 with the following lemma which is proven in the appendix:



Lemma 3.3. *Assume x^0 has $e(x^0) < 1$ and J is the minimal cut of $L(x^0)$. Then for all x with $e(x, J) \geq 1$,*

$$\|L(x) - L(x^0)\|_1 \geq 2(1 - e(x^0))$$

The cutting plane process continues until a solution x^k with $e(x^k) > 1 - 10^{-6}$ is obtained, or until a maximal number of iterations (50 in our implementation) is reached. We denote by $\mathcal{J} = \{J_0, J_1, \ldots, J_k\}$ the collection of cutting plane cuts, $J \subset \mathcal{V}$, found so far, and by $P_{\text{top}}(\mathcal{J})$ the linear program solved with these cutting plane constraints. If the algorithm is terminated before convergence we add algebraic connectivity constraint (10) to $P_{\text{top}}(\mathcal{J})$ and solve again to obtain our final lower bound. Before we solve we first check the value of $\lambda_2(x^k)$ and if it is larger that $\bar{\lambda}$ there is no need to solve, we already have the optimal solution at hand. We note that almost always the number of linear programs solved until the stopping criterion is reached is small (*i.e.*, up to 10 iterations), and the algebraic connectivity is indeed larger than $\bar{\lambda}$. This means that we are able to solve P_{top} using only a small number of linear programs, and without solving any expensive semi-definite programs.

The relaxation chain algorithm is summarized in Algorithm 1. Note that we allow to input the algorithm a set of predefined cuts; this is used in the branch and bound algorithm explained next.

Algorithm 1: Relaxation chain

Input: The linear energy to be minimized;
initial set of cuts $\mathcal{J}$ (default is $\mathcal{J} = \emptyset$).

set $k = 0$ stop=false;
while *(not stop) and ($k < 50$)* **do**
  $k = k + 1$;
  $x^k = \text{argmin}\, P_{\text{top}}(\mathcal{J})$;
  compute J_k the global min-cut of $G(x^k)$;
  $\mathcal{J} = \mathcal{J} \cup \{J_k\}$;
  if $e(x^k, J_k) \geq 1 - 10^{-6}$ **then**
    stop=true;

if $\lambda_2(x^k) < \bar{\lambda}$ **then**
  $x^k = \text{argmin}\, P_{top}(\mathcal{J})$ with (10);

Output: $x = x^k$ and cutting planes $\mathcal{J}$.

Energy. Since (2) is a combinatorial problem there could be a large number of feasible solutions. We would like to pick one that is in some sense "the best". For example, the minimal/maximal area solutions $S(x)$. This will be formulated as a linear energy in x:

$$E(x) = \sum_{ij} x_{ij} \,\text{area}(S_{ij}).$$

In our experiments we used this energy as well as two other energies which will be described in the results section.

Branch and bound. In several cases the relaxation chain described above returns an integer solution, which is thus the optimal solution of Problem 1. However to guarantee an optimal solution for Problem 1 in all cases we use a branch and bound approach, see Algorithm 2.

We begin with solving the relaxation chain as described in Algorithm 1 with $\mathcal{J} = \emptyset$, to obtain a solution $x(0)$ which minimizes $P_{\text{top}}(\mathcal{J}_0)$ with energy LB_0, which is a lower bound to the optimal solution of Problem 1. We round this solution to an integer solution $\bar{x}(0)$ and compute its energy to obtain an upper bound UB to Problem 1 (if $\bar{x}(0)$ doesn't fulfill the constraints then we set $UB = \inf$). If $LB_0 = UB$, as indeed happens quite often, then $\bar{x}(0)$ is the optimal solution and we can terminate the algorithm. Otherwise, we select a cell C_i for which the probability vector $x_i(0)$ has maximal entropy, and branch on cell i. That is, solve the relaxation chain again c_i times, with the additional constraint that $x_{ij} = 1$, $1 \leq j \leq c_i$, and starting from the set of cuts $\mathcal{J}_0$, to obtain new solutions $x(1), \ldots, x(c_i)$ and lower bounds $LB_1, \ldots, LB_{c_i}$. We also compute an upper bound for each solution by rounding and updating UB if the new upper bound is lower than all previous upper bounds. Now for each $1 \leq j \leq c_i$ we check whether $LB_j \geq UB$. If so then we can stop investigating all solutions with $x_{ij} = 1$. Otherwise, we choose another cell $C_{i'}$ and solve $c_{i'}$ problems to obtain solutions $x(j, j')$, $1 \leq j' \leq c_{i'}$ obtained by solving the relaxation chain while setting $x_{ij}, x_{i'j'} = 1$.

The branch and bound algorithm is guaranteed to achieve the global optimum of Problem 1. In general, this can occur only after all possible $\prod_{i=1}^{n} c_i$ possibilities were explored. The efficiency of the

Algorithm 2: Branch and Bound

Input: Linear energy to be minimized.

$UB = +\infty$, $node = \text{zeros}(1, n)$, $x_{opt} = NaN$
$\mathcal{J}_0 = \emptyset$, list = $\{node, \mathcal{J}_0\}$;
while list *isn't empty* **do**
  $(node, \mathcal{J}_0)$=pop(list);
  Solve relaxation chain initialized by $\mathcal{J}_0$ and fixing non-zero *node* coordinates to attain a partition list $\mathcal{J}$ and $x = \text{argmin}\, P_{top}(\mathcal{J})$;
  $LB = E(x)$;
  $\bar{x} = \text{round}(x)$;
  if *($\bar{x}$ is feasible) and* ($E(\bar{x}) < UB$) **then**
    $UB = E(x)$;
    $x_{opt} = \bar{x}$;
  if $LB \geq UB$ **then**
    do nothing (no need to explore this node).
  else
    Find cell C_i for which $x(i)$ has maximal entropy;
    for $j = 1 : c_i$ **do**
      $node(i) = j$;
      push(list,{*node*,$\mathcal{J}$})

Output: x_{opt}, the global minimizer of Problem 1.

algorithm depends on the successfulness of the convex relaxation in producing high quality lower bounds which can eliminate large branches of the tree of possibilities without exploring them. As our experiments will show, our algorithm is often (but not always) able to obtain global solutions in reasonable time.

Graph reduction. In every stage of the branch and bound algorithm, several entries of x are fixed, which corresponds to a fixed selection of surface patches for some of the cells. As a result, certain vertices of the graph $G(x)$ may now be already connected by the fixed surface patches. We can therefore merge each known connected component of the graph to a single vertex, to obtain a new, smaller graph which is connected if and only if the previous graph, with the known fixed edges, is connected.

4 RESULTS

4.1 Reconstruction from cross sections

We applied our BnB (branch and bound) algorithm to the problem of surface reconstruction from cross sections, and compared it with the DP algorithm of [Zou et al. 2015], and with its extended version for multi-labeled materials from [Huang et al. 2017]. In all problems we used the cell partition, surface patches, and linear energy provided by the algorithm of [Huang et al. 2017; Zou et al. 2015], and we only compared the optimization method. For two-labelled domains, the surface patches are constructed as level sets of a scalar function at a set of scalar levels chosen to yield different surface topologies. For multi-labelled domains, the level sets are replaced by so-called

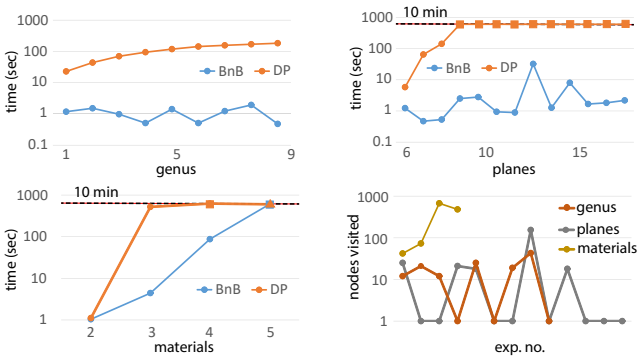


Fig. 2. Running time comparison of BnB and DP algorithms, as a function of the number of planar cross sections, the number of materials, and the genus of the surface. Both algorithms were terminated after ten minutes if they did not converge. Square labels mean the algorithm returned a feasible, but not necessarily globally optimal solution. Triangle label means the algorithm did not find a feasible solution in the allotted ten minutes. In contrast with DP, the timing of BnB is not significantly affected by a large number of intersecting planes, or high genus problems. Large multi-label problems are difficult for both algorithms, but BnB is able to achieve a feasible solution for these problems while DP runs out of memory.

interface sets of a vector function, which are surface networks parameterized by vectors. The inset shows a candidate surface patch for the chicken heart reconstruction problem we discuss below. Note that the surface patch has nontrivial topology- it is non-contractible and has two connected components. Our method is readily extended to the multi-label surface reconstruction scenario by imposing the genus and connectivity constraints on each constrained sub-surface separately.

Synthetic data. We generated synthetic examples to evaluate the dependence of the algorithms' complexity on genus, the number of cross sections, and the number of labels. Our results are shown in Figure 2. Our algorithm converged in a few seconds for the problems with changing genus and number of planes, often several orders of magnitude faster than DP which deteriorates in terms of computation time and memory usage as genus or plane number is increased. For the changing number of planes and genus we set 10 minutes as a time limit, and DP reached this limit in the changing number of planes experiment without achieving an optimal solution.

For problems with changing numbers of materials, both algorithms encountered difficulties as the number of materials was increased. BnB was significantly faster for problems with 2-4 materials. For five materials both algorithm reached the ten minute limit, but BnB was able to return a feasible solution while DP did not find a feasible solution in the alloted time. For six materials (not shown in the graph) both algorithm could not even find a feasible solution in ten minutes, but BnB could find a feasible solution within 18.5 minutes while DP ran out of memory after running for two hours.

Figure 4 shows two examples from the multi-material problem set: (a) and (b) shows the input for the 4 and 6 material examples; and (c)-(d) show their reconstruction by BnB. In this case the 4 material problem was solved in less than two minutes, while DP was stopped

Sillicon reconstruction					
Planes	Genus	Parallel?	BnB (s)	DP (s)	energy gap
25	93	yes	43.16	446.51	0
33	124	yes	66.95	1729.72	0
41	155	yes	2.72	588.09	0
49	189	yes	602.65	636.45	5.00E-04
57	217	yes	603.21	571.34	2.20E-04
65	248	yes	422.9	615.64	0
9	31	no	0.25	267.38	0
9	free	no	1.31	294.54	0

Table 1. Timing comparison of the BnB and DP algorithms for reconstruction of the crystal structure of silicon.

after ten minutes without achieving an optimal solution. The energy gap (normalized difference between energy) between their solution and the optimal result achieved by our solution was 0.72. Figure 4 (d) shows the feasible, sub-optimal solution found by BnB for the 6 material problem. As mentioned above DP was not able to find a feasible solution for this problem.

Real-life data. We ran our algorithm on the real life examples from [Zou et al. 2015], and on the multi-labeled examples from [Huang et al. 2017]. In all these examples, both DP and BnB converge in less than a second. Our algorithm finds the correct integer solution after only one linear program solve. We note that in [Huang et al. 2017] DP was reported to solve the chicken heart reconstruction problem in 56 seconds, but in fact this relatively slow solution was due to the fact that a suboptimal method for traversing the cells was selected in the implementation of their algorithm. Once this error was repaired DP solves this problem as well in less than a second.

We applied our algorithm to cross-sections of the crystal structure of silicon, which has a diamond cubic crystal structure (consists of two inter-penetrating face centered cubic lattices). This data is more challenging due to its high genus and large number of contours and variables. Figure 3 shows an example of the input data (top) and our reconstructed surfaces (bottom). We compare timing and objective values with DP in Table 1 for several different silicon data inputs with varying number of planes (parallel and non-parallel) and target genus. In most examples parallel planar cross sections were used, while in the last two experiments in the table the cross sections were non-parallel (*e.g.*, Figure 3, right). In all examples when a global solution was found by our algorithm it was faster than DP, sometimes by two orders of magnitude. The difference between the methods was especially significant for non-parallel examples, where BnB converged within a second while DP took more than 250 seconds. Both algorithms were stopped if they ran for (approximately) ten minutes, and in this case the best objective value obtained up to this point was reported. BnB reached the time limit twice, and DP reached the time limit in five examples. In the examples where both BnB and DP reached the time limit the energy obtained by DP was marginally better (the normalized difference between energies was less that 10^{-3}). We note that when both algorithms converge the energy values are always the same since both algorithm are guaranteed to solve Problem 1 globally.

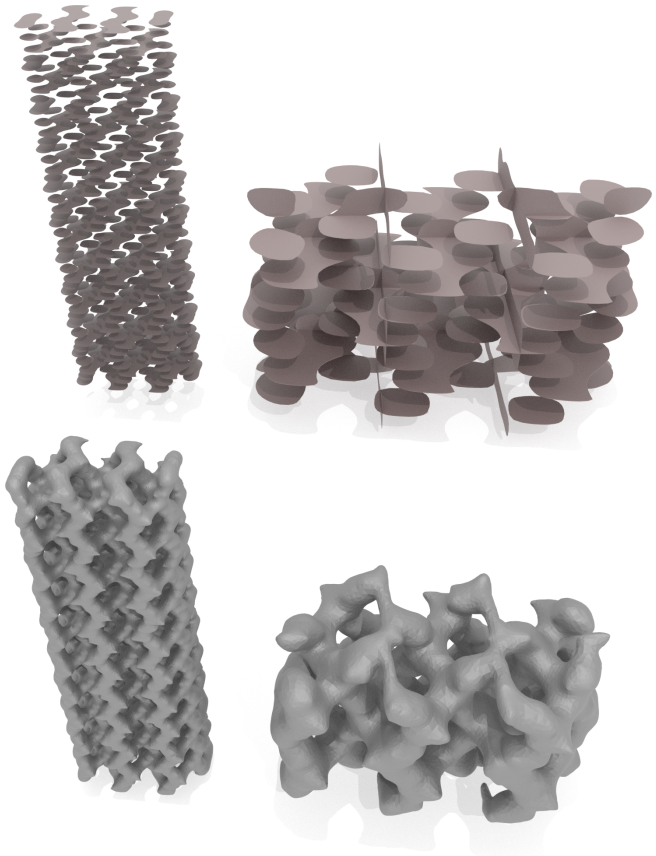


Fig. 3. Reconstruction of the crystal structure of silicon (bottom) from its cross section (top). The bottom left model has genus 167 while the bottom right has genus 31.

We then applied our algorithm to a stack of cross-sections of a 4-week old corn root imaged by CT. This data is particularly challenging due to the large number of cross sectional curves: 317 cross sections on 26 parallel planes. The energy we used is the minimal total length of the branches. Figure 1 shows the reconstructed surface (right) as well as the cross section data (left). Solving this problem took 103 seconds. Figure 5 (a) shows an overlay of all the possible connections of cross sections; (b) shows the globally optimal reconstruction result with a genus zero constraint; (c) shows the unconstrained solution (*i.e.*, solving (11) without the topological constraints), note the disconnected branch and loops; (d) shows the optimal reconstruction with genus 20.

4.2 Isosurface reconstruction

Isosurface reconstruction deals with the problem of finding the zero level set of a smooth function $f : \mathbb{R}^3 \to \mathbb{R}$. This level set is indeed a surface provided that zero is a regular value of f. We assume the zero level set of f is bounded in the unit cube $(0, 1)^3$, and partition the unit cube into a grid of cubes (cells in our former terminology).

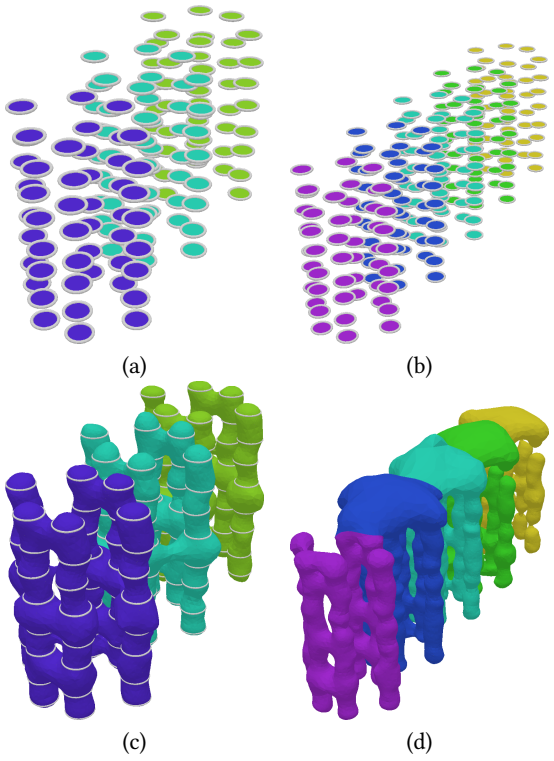


Fig. 4. Multi-material synthetic reconstruction: (a) and (b) show cross section input for 4 and 6 material reconstruction problems, respectively; (c) and (d) shows the reconstructions found by our algorithm; (c) is the optimal solution while (d) is a feasible, not optimal solution.

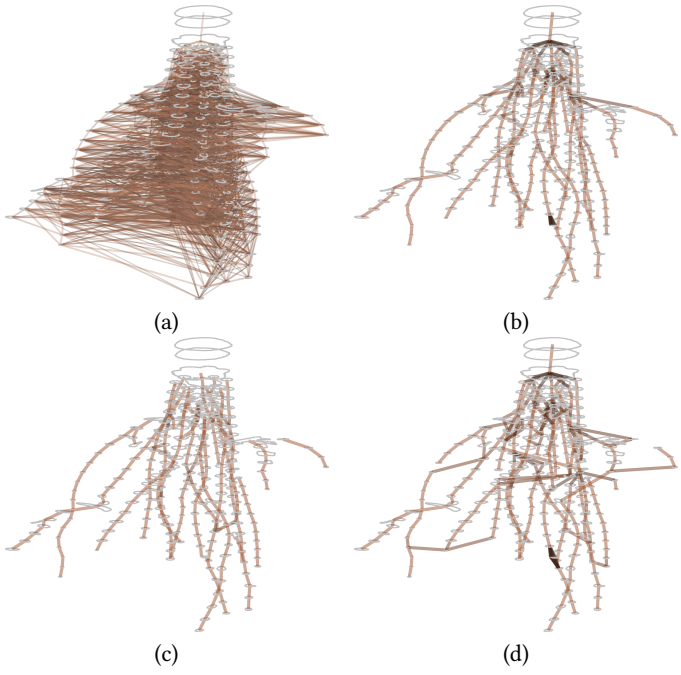


Fig. 5. Reconstruction of corn roots from cross sections. (a) shows all possibilities (in brown) of connecting the cross section contours (in silver). (b) shows our globally optimal genus zero connected solution. (c) shows the non connected solution obtained from the unconstrained linear relaxation (*i.e.*, best solution in each cell), and (d) shows a genus 20 reconstruction of the corn roots.

We are given an evaluation of f on the vertices of the cubes, and our goal is to reconstruct $f^{-1}(0)$ from this information. The celebrated marching cube (MC) algorithm and its variants solve this problem locally in each cube, without taking into account global information such as topological constraints. For a given sign configuration on the vertices of a cube, MC assigns a unique surface patch. While this approach is very successful for densely sampled functions, it may cause topological errors for sparsely sampled functions. In contrast, our approach can find a globally correct solution, providing that the family of surface patches per cells is rich enough to allow for a correct global solution. Our method for choosing surface patches per cell is described in the appendix.

In figure 6 we show two examples where our algorithm achieves a topologically correct result and MC does not. On the left we show a torus with a small handle, which the MC algorithm "doesn't notice". We note that in this example the MC algorithm will attain a topologically correct solution if the resolution of the cube partition is increased. The remainder of figure 6 show reconstructions produced by the MC algorithm and the BnB algorithm of a coned surface with two cone points, at three different resolutions. It can be easily shown that *at any resolution* the MC algorithm will fail to achieve a connected solution, since the intersection of the surface with cubes sufficiently close to the cone point will result in a sign configuration which causes MC to choose a disconnected solution.

In contrast to the MC algorithm, our algorithm reconstructs both surfaces correctly by allowing for additional topological possibilities per cell and forcing genus and connectivity constraints. For the highest resolution surface on the far right of Figure 6 our reconstruction took 3 seconds while MC took half a second. For the other three surfaces in the figure MC was faster as well, although both algorithms required less than a second.

5 CONCLUSION AND FUTURE WORK

We devise an algorithm for solving the topology-aware surface reconstruction problem. Our key insight is that the topological constraints can be formulated as convex constraints over the unknown integer variables. In turn, when these variables are relaxed a tight convex relaxation is achieved. We use the relaxation in a branch and bound framework to solve the reconstruction problem, often after a small number of steps and several order of magnitude faster than competing methods.

The main limitation of our approach is that each computation of a lower bound requires solving (in the worst-case) up-to 50 linear programs and a semidefinite program. Although the lower bound can be tight and provide the solution very fast, we have found instances, *e.g.*, five material problem (see Figure 2, bottom left) where we were not able to find an optimal solution in reasonable time.

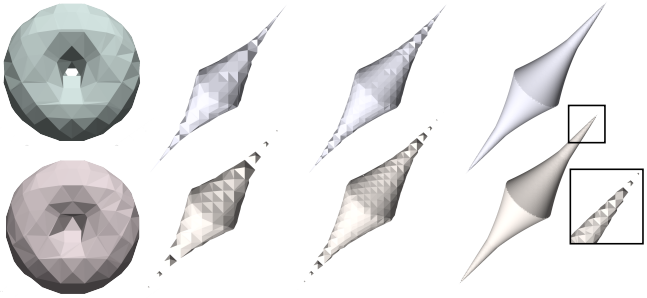


Fig. 6. Isosurface reconstruction by BnB (top) and MC (bottom). BnB correctly reconstructs both surfaces, while MC returns a surface with incorrect genus (left) and a disconnected surface (next three surfaces, which show the same surface reconstructed at three different resolutions). For the second surface the MC algorithm will return a disconnected surface, regardless of the resolution considered.

One very interesting future work direction is to incorporate quadratic energies into this framework. Quadratic energies can be used to encourage smooth output surfaces and other pairwise penalty terms. Another direction for future work is topological reconstruction of surfaces with $k > 1$ connected components.

Acknowledgements. We would like to thank the anonymous reviewers for their helpful comments. RL, ND, YK and YL acknowledge the support of the European Research Council, ERC-cog grant no. 771136-LiftMatch, and the Israel Science Foundation, grant no. ISF 1830/17. ZH and TJ acknowledge the support of NSF grants IIS-0846072, IIS-1302200, RI-1618685.

REFERENCES

Omid Amini, Jean-Daniel Boissonnat, and Pooran Memari. 2013. Geometric tomography with topological guarantees. *Discrete & Computational Geometry* 50, 4 (2013), 821–856.

Marco Attene, Marcel Campen, and Leif Kobbelt. 2013. Polygon mesh repairing: An application perspective. *ACM Computing Surveys (CSUR)* 45, 2 (2013), 15.

Gill Barequet and Amir Vaxman. 2009. Reconstruction of Multi-Label Domains from Partial Planar Cross-Sections. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 1327–1337.

Pierre-Louis Bazin and Dzung L Pham. 2007. Topology-preserving tissue classification of magnetic resonance brain images. *IEEE transactions on medical imaging* 26, 4 (2007), 487–496.

Amit Bermano, Amir Vaxman, and Craig Gotsman. 2011. Online reconstruction of 3D objects from arbitrary cross-sections. *ACM Transactions on Graphics (TOG)* 30, 5 (2011), 113.

Jean-Daniel Boissonnat and Pooran Memari. 2007. Shape reconstruction from unorganized cross-sections. In *Symposium on Geometry Processing*. 89–98.

Evgeni Chernyaev. 1995. *Marching cubes 33: Construction of topologically correct isosurfaces.* Technical Report.

Fan RK Chung. 1997. *Spectral graph theory.* Number 92. American Mathematical Soc.

Paolo Cignoni, Fabio Ganovelli, Claudio Montani, and Roberto Scopigno. 2000. Reconstruction of topologically correct and adaptive trilinear isosurfaces. *Computers & Graphics* 24, 3 (2000), 399–418.

Lis Custodio, Tiago Etiene, Sinesio Pesco, and Claudio Silva. 2013. Practical considerations on Marching Cubes 33 topological correctness. *Computers & Graphics* 37, 7 (2013), 840–850.

Arindam Kumar Das and Mehran Mesbahi. 2005. K-node connected power efficient topologies in wireless networks: a semidefinite programming approach. In *Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE*, Vol. 1. IEEE, 6–pp.

Bruno Rodrigues De Araújo, Daniel S Lopes, Pauline Jepp, Joaquim A Jorge, and Brian Wyvill. 2015. A survey on implicit surface polygonization. *ACM Computing Surveys (CSUR)* 47, 4 (2015), 60.

Miroslav Fiedler. 1973. Algebraic connectivity of graphs. *Czechoslovak mathematical journal* 23, 2 (1973), 298–305.

Arpita Ghosh and Stephen Boyd. 2006. Growing well-connected graphs. In *Decision and Control, 2006 45th IEEE Conference on.* IEEE, 6605–6611.

Zhiyang Huang, Ming Zou, Nathan Carr, and Tao Ju. 2017. Topology-controlled Reconstruction of Multi-labelled Domains from Cross-sections. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 76.

Tao Ju, Qian-Yi Zhou, and Shi-Min Hu. 2007. Editing the Topology of 3D Models by Sketching. *ACM Trans. Graph.* 26, 3, Article 42 (July 2007). https://doi.org/10.1145/1276377.1276430

Lu Liu, C. Bajaj, Joseph Deasy, Daniel A. Low, and Tao Ju. 2008. Surface Reconstruction From Non-parallel Curve Networks. *Comput. Graph. Forum* 27, 2 (2008), 155–163. http://dblp.uni-trier.de/db/journals/cgf/cgf27.html#LiuBDLJ08

William E Lorensen and Harvey E Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. In *ACM siggraph computer graphics*, Vol. 21. ACM, 163–169.

Jing Qian, Venkatesh Saligrama, and Yuting Chen. 2014. Connected sub-graph detection. In *Artificial Intelligence and Statistics*. 796–804.

Andrei Sharf, Thomas Lewiner, Ariel Shamir, Leif Kobbelt, and Daniel Cohen-Or. 2006. Competing fronts for coarse–to–fine surface reconstruction. In *Eurographics*. Vienna, 389–398. http://www.mat.puc-rio.br/~tomlew/competing_fronts_eg.pdf

Andrei Sharf, Thomas Lewiner, Gil Shklarski, Sivan Toledo, and Daniel Cohen-Or. 2007. Interactive topology-aware surface reconstruction. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 43.

Mechthild Stoer and Frank Wagner. 1997. A simple min-cut algorithm. *Journal of the ACM (JACM)* 44, 4 (1997), 585–591.

Francisco Velasco, Juan Carlos Torres, Alejandro León, and Francisco Soler. 2008. Adaptive cube tessellation for topologically correct isosurfaces. *JVRB-Journal of Virtual Reality and Broadcasting* 5, 3 (2008).

Zoë Wood, Hugues Hoppe, Mathieu Desbrun, and Peter Schröder. 2004. Removing excess topology from isosurfaces. *ACM Transactions on Graphics (TOG)* 23, 2 (2004), 190–208.

Kangxue Yin, Hui Huang, Hao Zhang, Minglun Gong, Daniel Cohen-Or, and Baoquan Chen. 2014. Morfit: interactive surface reconstruction from incomplete point clouds with curve-driven topology and geometry control. *ACM Trans. Graph.* 33, 6 (2014), 202–1.

Yun Zeng, Dimitris Samaras, Wei Chen, and Qunsheng Peng. 2008. Topology cuts: A novel min-cut/max-flow algorithm for topology preserving segmentation in N–D images. *Computer vision and image understanding* 112, 1 (2008), 81–90.

Shizhe Zhou, Changyun Jiang, and Sylvain Lefebvre. 2014. Topology-constrained synthesis of vector patterns. *ACM Trans. Graph.* 33, 6 (2014), 215–1.

Ming Zou, Michelle Holloway, Nathan Carr, and Tao Ju. 2015. Topology-constrained surface reconstruction from cross-sections. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 128.

Ming Zou, Tao Ju, and Nathan Carr. 2013. An algorithm for triangulating multiple 3D polygons. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 157–166.

6 APPENDIX

Proof of Theorem 3.1. Assume $G(x)$ is connected, we want to show that $S(x)$ is connected. By construction of $G(x)$, there is a path in $S(x)$ connecting any two segments γ_i and γ_j. Given an arbitrary point in $S(x)$, it is a member of one of the surface segments S_{ij}, and thus by our requirements on the surface segments must be connected to one of the surface segments γ_i. Thus $S(x)$ is connected.

Now assume $S(x)$ is connected, we want to prove $G(x)$ is connected. We choose arbitrary vertices in the graph corresponding to the surface segments γ_0, γ_1, we want to show that they are connected by the edges of $G(x)$. We begin with a path $\beta : [0, 1] \to S(x)$ connecting $p \in \gamma_0$ and $q \in \gamma_1$, whose existence is guaranteed by the connectivity of $S(x)$. Next we choose a minimal partition $0 = t_0 < t_1 < \ldots t_k = 1$ of the unit interval, such that the restriction of β to each subinterval is contained in a single surface patch. By the minimality of the partition, each $\beta(t_i)$, $0 < i < k$ belongs to the intersection of two surface patches $S_1 \cap S_2 \subset C_1 \cap C_2$ in the common boundary of the cell C_1 containing $\beta(t_{i-1}, t_i)$ and a distinct cell C_2 containing $\beta(t_i, t_{i+1})$. The assumption on the surface patches interpolating the segments Γ implies that in each connected components of $S_1 \cap S_2$ there is at-least a part of some segment which we denote by $\gamma_{t_i} \in \Gamma$. Therefore by construction of G, γ_0 is connected to γ_{t_1}

which is connected to γ_{t_2} and so forth until $\gamma_{t_{k-1}}$ which is connected to $\gamma_{t_k} = \gamma_1$.

□

Proof of Theorem 3.2. Part (1) of the theorem follows directly from (9) which is valid also for weighted graphs.

For part (2) of the theorem, note that $\bar{\lambda} = f(\pi m^{-1})$ for

$$f(x) = 2(1 - \cos(x)) \leq x^2.$$

Therefore $\bar{\lambda} = f(\pi m^{-1}) \leq cm^{-2}$ for $c = \pi^2$. Now consider $\bar{L}$ defined in (10); it can be seen as the Laplacian of a full graph. Direct computation shows that $\lambda_2(\bar{L}) = \bar{\lambda}$ and

$$e(\bar{L}) = \bar{\lambda}\frac{m-1}{m} \leq cm^{-2}.$$

For part (3) of the theorem, we choose for any given L_0 a new laplacian $L = L_0 + \bar{L}$. Then

$$\lambda_2(L) = \min_{v|\ \|v\|=1,\ v\perp\mathbf{1}} v^T L v \geq \min_{v|\ \|v\|=1,\ v\perp\mathbf{1}} v^T \bar{L} v = \lambda_2(\bar{L}) = \bar{\lambda}$$

and

$$\|L - L_0\|_1 = \|\bar{L}\|_1 = \bar{\lambda}2(m-1) \leq 2cm^{-1}$$

□

Proof of Lemma 3.3. For x, x^0 and J satisfying the conditions of the lemma we have

$$\left\|L(x^0) - L(x)\right\|_1 \geq 2 \sum_{j\in J,\, i\notin J} \left|L_{ji}(x^0) - L_{ji}(x)\right| \tag{12a}$$

$$\geq 2 \left| \sum_{j\in J,\, i\notin J} \left(L_{ji}(x) - L_{ji}(x^0)\right)\right| \tag{12b}$$

$$= 2\left|e(x,J) - e(x^0)\right| \geq 2(1 - e(x^0)) \tag{12c}$$

□

Surface patch selection for isosurface reconstruction. For every sign assignment to a cube's vertices, we select a number of possible surface patches, which are constructed in three stages as follows:

Stage (0): We define the surface patches uniquely on the edges of the cube: On each edge of the cube, we add a point to the candidate surface patch if the two vertices of the edge are assigned opposite signs.

Stage (1): We define the surface patches on each of the six faces of the cubes. Each face has been assigned 0, 1, 2 or 4 points to its four edges in stage (0). If a face contains two points then they are connected by an edge, and if the face contains four points then we connect two pairs of points by an edge each. This results in an ambiguity: There are two ways to select these edges so that they do not intersect. Thus for each face we have two possible interpolations (if there are four points in the face), or a unique interpolation (if the face contains two vertices or less).

Stage (2): At the conclusion of stage (1) we have one or two candidates per face, and so in total up to 2^6 reconstruction options for the boundary of the cube. For each fixed reconstruction on the boundary, we find the connected components $B_1, \ldots, B_k$ of the reconstruction. We then consider all possibilities of connecting $B_1, \ldots, B_k$. Each connection possibility partitions the k sets into $j \leq k$ connected components, and the total number of possibilities is the number of partitions of the set $\{1, \ldots, k\}$, which is known as the k^{th} *bell number*. This number is double exponential in k, however for any fixed reconstruction of the boundary k will never be larger than four, and thus the bell number is bounded by fifteen. This is because in Stage (0) at most twelve points are added to the surface, and each B_i will contain at least three of these points. Once a connection scheme is selected, we complete it to a surface patch using the triangulation algorithm of [Zou et al. 2013].